DATA CONVERSION:

The conversion of MicroStation cells to Arc/Info Symbology

by

Katherine E. Riegelmann and James R. Hoff
DYNTEL Corporation
3530 Manor Dr. Suite 4
Vicksburg, Mississippi
ph: (601) 634-2468, fax (601) 634-4584
email: riegelk@ex1.wes.army.mil

and

Edward A. Riegelmann
USAE Waterway Experiment Station
Tri-Service CADD/GIS Technology Center
3909 Halls Ferry Road
Vicksburg, Mississippi 39180
ph: (601) 634-2468, fax (601) 634-4584
email: riegele@ex1.wes.army.mil

THE CONVERSION OF MICROSTATION CELLS TO ARC/INFO SYMBOLOGY

Introduction

The Tri-Service Spatial Data Standards (TSSDS)[1] have been designed and implemented by the Tri-Service CADD/GIS Technology Center at the U.S. Army Corps of Engineers' Waterways Experiment Station, in Vicksburg, Mississippi. The TSSDS were developed as a comprehensive master and environmental planning data model for Air Force, Army, and Navy installation projects. The standards address data content, classification, format, and presentation of geospatial entities depicted in large scale mapping (primarily, 1 inch = 400 feet (1:4800) to 1 inch = 50 feet (1:600)). The TSSDS development program has defined a generic data model, based on the Spatial Data Transfer Standards (SDTS - FIPS 173), that can be implemented on multiple CADD/GIS platforms.

The need for standardization arises from the CADD/GIS community's need to share data, such as drawings, attribute information and symbology, across CADD/GIS platforms. The TSSDS provide a variety of standardized symbology to be used across multiple platforms. These symbols were designed and created in *MicroStation* and stored in a variety of cell libraries as they relate to the TSSDS. To avoid duplication of efforts, these symbols were converted from *MicroStation* to *AutoCAD* and *Arc/Info* formats, respectively. And, as a result, the symbols had to be customized to meet the *AutoCAD* and A*rc/Info* requirements. The conversion of symbology from one platform to another is a subject that has not been widely addressed by the CADD/GIS industry and user community. The intent of this paper is to provide guidance to the CADD/GIS community on the conversion of symbology from a M*icroStation* cell library format to an *Arc/Info* markerset format.

Methodology

MicroStation[2]

MicroStation is a Computer-Aided Design and Drafting (CADD) computer system that is used to input, store, retrieve, analyze, manipulate, and present graphic and non-graphic data. *MicroStation* is used in civil, mechanical, and other engineering design and architectural activities. This system has interactive graphics capabilities and includes a variety of engineering calculation and analysis functions.

MicroStation design (*.dgn)* files, also referred to as IGDS (Interactive Graphic Design Software) files, may contain any of 127 possible element types. ***IGDSARC***[3], the *Arc/Info* command that initiates the conversion of IGDS files, only translates 11 of these 127 element types. Translatable elements include: cell (2), line (3), line string (4), text (17), shape (6), text node (7), curve (11), complex string (12), complex shape (14), ellipse (15), and arc (16). The number beside the element type is used as a translation parameter in *Arc/Info's* ***IGDSARC*** command. An IGDS file can have one, many, or all of these element types. IGDS elements are organized by layer or levels. Any number of levels can be extracted to produce a coverage within *Arc/Info* based on one, many, or all of these elements.

Cell Definition

Within *MicroStation*, elements can be grouped into a single element called a cell. Cells are what *Arc/Info* would refer to as symbols or markers. A ***cell*** is a group of elements combined into one complex element that describe a symbol. Cells are stored in a cell library for use within the existing design file as well as any other design file referencing the cell library. Cells are used to minimize redundancy, and standardize the use of frequently placed symbols.

Cell Extraction from MicroStation

To place cells from a cell library into a *MicroStation* design (*.dgn)* file, there is a utility available from the Intergraph Bulletin Board System called Cellplot.ma. Once all cells are placed in a single design (*.dgn*) file, the user can create a design (*.dgn)* file for EACH cell, making the conversion to a symbol in *Arc/Info* much easier.

In *MicroStation*, open the design (*.dgn*) file that holds all the cells from the cell library. To extract a cell one at a time, use the ***PLACE BLOCK***[2] command. This command allows the user to put a "block" around the cell of choice. *Arc/Info* will read the boundaries of this "block" and produce a TIC and BND file based on those boundaries. To maintain the integrity of the cell, it is necessary to place a "block" that is equal on each side, i.e., x = y. There is a command in *MicroStation* called ***DX***[2]***=****x,y*, i.e dx = 1,-1 that allows the user to place a "block" equally on each side. Once the first point is placed, this command can be executed to give the user a "square block". To maintain consistent scaling of the individual symbols, use the same "block" dimensions as a template for all symbols, which will ensure the consistency of the symbol scale between platforms.

It is important to note that if the cell contains text, (*MicroStation* element 17), and the user intends on using that text as part of a symbol, the user must breakdown the text into primitive elements, i.e. line (3), linestring

(4), shape (6), etc., to better manipulate them within the *Arc/Info* coverage. *Arc/Info* WILL translate element text (17) and store it in a sub-class called anno.igds within a coverage. The text will be placed in the anno.igds sub-class of the coverage using the ANNOTEXT option during the use of the ***IGDSARC***[3] command (see page 5). However, to store text within an *Arc/Info* font library, anno.igds will NOT be read. So, it becomes necessary to breakdown the cell into its primitive elements so the text can be manipulated as a linestring (4). The ***DROP COMPLEX SHAPE***[2] command will breakdown the elements as stated above. If the cell does not contain text, the cell will be saved within the design (*.dgn)* file as element (2), and *Arc/Info* will translate the cell components to arcs.

If a cell is broken down into its primitive elements and text (17) exists, the ***DROPTEXT***[2] command can be executed to decompose the text into linestrings (4). Now a design (*.dgn)* file can be created with the original cell's primitive elements. This will allow *Arc/Info* to read the coverage as arcs without text.

Note: Text can also be added to the symbol itself within the *ArcPlot* session when creating the markerset. The text would reside within another layer of the marker, making reference to a default textset or a custom designed textset. How to manage the text is a user preference based upon an individual's plans of how to organize the font libraries and the individual patterns.

Now that the "block" has been placed, execute the ***DROP COMPLEX SHAPE*** and ***DROP TEXT*** as required. Next, isolate the elements that were "blocked" off so individual design (*.dgns)* files can be produced for each symbol. The ***PLACE FENCE***[2] command creates a rectangle that is used to select a set of elements. Once the user has defined the contents of the "fence", it is then necessary to ACCEPT the "fence" contents and create a new design (*.dgn*) file. The user will be prompted to accept or reject the contents of the fence. The user can place this new design (*.dgn)* file in the directory of choice, i.e., ***ff***[2]*=<path>/<file name>.dgn.* The resulting design (*.dgn)* file can now be converted to an *ARC/INFO* coverage.

IGDS to ARC

Once the design (*.dgn)* file has been created, the ***IGDSINFO***[3] command can be executed from the *ARC* prompt. This command will break down the elements of the design (*.dgn)* file and give the user a better understanding of only those elements needed to create a customized coverage. This command provides a complete summary of element types; the number of elements in each level, color, line style and weight, and how often that element occurs in the file. Unsupported elements will be flagged NOT SUPPORTED.

IGDSINFO *<igds_file>*

After determining the structure of the IGDS design (*.dgn)* file, convert the design (.dgn) file from *MicroStation* to *Arc/Info* by using the ***IGDSARC*** command at the *Arc* prompt. See below.

Arc: IGDSARC *<in_igds_file> <out_cover> {2D|3D} {xmin} {ymin} {xmax} {ymax} {OVERLAP | INSIDE} {text_width}*

Usage for this command:

<in_igds_file> - name of IGDS file to be translated.

<out_cover> - name of the coverage to be created from the input IGDS file. The <out_cover> may contain any combination of points, lines or annotation. An INFO file name <out_cover>.ACODE will be created for IGDS attributes. An optional binary system file called <out_cover>.alink will contain database pointers.

{*2D | 3D*} - specifies IGDS file to be two or three dimensional.

2D - two dimensional IGDS file (default).

3D - three dimensional IGDS file.

{xmin} {ymin} {xmax} {ymax}

- desired coordinate range of elements to be translated. Either all or none of these values must be entered. The default is the entire range of the IGDS file. IGDSINFO will display the x,y range of the file.

{*OVERLAP | INSIDE}* - when x and y range is specified.

OVERLAP - all elements whose bounding boxes overlap the specified x,y range are extracted.

INSIDE - all elements whose bounding boxes are completely inside the specified x,y range are extracted.

{*text_width}* - width of the IGDS-TEXT item. The default will be 12 if no text_width is given.

Upon entering the ***IGDSARC*** command, a dialog box will appear and prompt the user for the IGDS-to-ARC conversion as follows:

Enter layer names and options (type END or $REST when done)
==========
Enter 1st layer and options:
Enter 2nd layer and options:
.
.

Here is the usage for the each response:

<layer_name | END | $REST> <level> <color> <line_style> <line_ weight> <element type> {cell_name} {option_1 ... option_n}

Wild cards (*) can be used in place of the above responses. The wild card indicates that all the elements associated with the chosen element will be translated provided they meet the extraction criteria. Note: the user cannot leave empty fields; an integer or wild card must be used.

Special Note: when entering layers and options the user has the following options:

<*layer_name* | *END* | *$REST*> - layer description.

layer_name - user defined

END - ends layer input.

$REST - allows the user to take all remaining layers that have been previously specified. Must be the last layer specified, END will not be needed. If $REST is not specified, remaining layers WILL NOT be translated. To translate ALL element types and include them in one coverage, use the ALL option i.e., $REST ALL.
Note: Typically, for the purposes of converting cells to a coverage, the $REST ALL function is adequate. It, again, is a user preference as to how to extract elements and store them

<*level*> - levels range between 1 and 63.

<*color*> - colors range between 0 to 255.

<*line*_style> - line styles range between 0 to 7.

<*line_weight*> - line weights are between 0 to 31.

<*element_type*> - the valid element types range from 1 to 127. Keep in mind that only 11 elements types are translated by Arc/Info, as stated previously.

{cell name} - there must be an entry here if the element type is 2; otherwise the field should be blank. If the $REST ALL option is used no entry needs to be made. The cell will be translated to arcs.

{*option_1...option_n*} - the options here are many depending on the users needs and desired output for the coverage**.**

Now that the coverage is created, it is not necessary to build topology for the coverage. Topology has no effect on the creation of FONTS, PATTERNS, and MARKERSETS, and would only be neccessary to perform spatial analysis on the coverage itself. Again, it depends upon the user's use of the coverage.

Resulting from the execution of the ***IGDSARC*** command is an *INFO* data file named <out_cover>.**ACODE**[3], which holds the graphic attribution codes from the IGDS design (*.dgn*) file. These codes can be used to perform relates and joins to the feature attribution tables as desired. The ACODE file can also be used to manage symbology and store marker characteristics such as fonts and pattern numbers. Within the ACODE file is a graphic attribution code called IGDS-TEXT. It is within that field that the cell name from IGDS design (*.dgn*) file cell (2) is stored.

Editing the Coverage in ArcEdit

Once the coverage has been created using the ***IGDSARC*** command, the coverage is ready to be edited within the *ArcEdit* session. When the coverage (symbol) is drawn in *ArcEdit*, the coverage appears with a "box" around the desired coverage. This "box" is the "block" that was placed around the symbol (cell) in *MicroStation.* This arc will need to be deleted unless it is part of the symbol itself.

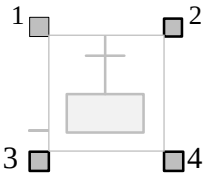
If there is annotation within the symbol and the ***DROP COMPLEX SHAPE*** and ***DROP TEXT*** commands were executed, the text will appear as arcs and can be edited and manipulated as such.

During the execution of the ***IGDSARC*** command, *BND* and *TIC* files were automatically created, based on the placement of the "block". However, the TIC ID's are not properly ordered when they appear in the *ArcEdit* session. Each individual coverage (symbol), needs to be prepared for insertion into a font library in the form of a pattern. During the *ArcEdit* session it is necessary to re-calculate the TIC ID's to their appropriate order prior to placing it in a font library. An error message will appear at the *Arc* prompt stating an error has occured if the TIC ID's are not properly calculated. *Arc/Info* does not allow TIC's to have the same value as another TIC at any time. The following commands allow the user to add a value of 10 to each TIC, making it easier to re-calculate the TIC's to their correct values:

```
EF TIC
SEL ALL
CALCULATE  $ID = $ID + 10
```

Now each TIC will be numbered 11, 12, 13 and 14 respectively, and editing is simplified. Now the TIC's can be calculated to their proper order.

Example:



Once the TIC's are calculated correctly, the arcs must ALSO be properly calculated to reflect the desired output. To create specific symbols, such as solid fill (also called panel fill) or single lines, it is necessary to calculate the User-ID's of the coverage arcs within the *ArcEdit* session. This will determine how the arcs will be stored in the font library and pattern. If the User-ID's are not calculated to a value listed below, the font will not be read. An error message will appear stating just that. Note: Within the *ArcEdit* session, if an arc is to be calculated for solid fill, the arc must have only one node. Select all the arcs and use the ***UNSPLIT***[3] command to eliminate pseudo nodes.

UNSPLIT *<none | all>*

If you want: (within the marker)	**The User-ID must equal:**
Single-lines	2
Fat-lines	6
Solid fill	7

Generalizing a Coverage

Because there are limitations to the amount of vertices that can exist within a font, it is necessary to use the *ARC* command ***GENERALIZE***[2] to weed out vertices within a coverage. A weed tolerance is based on the amount of weed necessary. The vertex limitation per pattern is 1,000. There are only a total of 39,000 vertices allowed per font. This is a MAJOR limitation when creating customized markers, especially when the user can have up to 127 PATTERNS per font. Coverages (patterns) that have a lot of circles, curved arcs and very dense arcs SHOULD be generalized. When a lot of markers are involved, and many fonts are being used, this command becomes very necessary!

GENERALIZE *<in_cover> <out_cover> {weed tolerance}*

Creation of Fonts and Patterns

Now that a coverage exists, the appropriate ID's have been calculated, and the coverage has been generalized (if necessary), the creation of fonts and patterns begins. In order for a marker to be read within a markerset, each marker must make reference to a font and a graphic pattern that represents each symbol. A font organizes and stores patterns that have similar applications and characteristics. A pattern is stored within a font as a pattern cell and holds graphic contents that are used in symbol layers.

In order for a marker to be drawn, the marker must exist in a font library and be assigned to a pattern. The pattern acts as a "coverage" within a font and can be manipulated as such. The ***ARCFONT***[2] command is used to insert a coverage into a font, and assign a pattern number. This is done within the *ARC* session.

Once a pattern is in place within the font library, the user may find it necessary to take that pattern back out and manipulate it. The ***FONTARC***[2] command allows the user to pull the pattern out of the font library as a coverage. After the changes have been made to the pattern, the ***ARCFONT*** command would be used again to return the pattern to the font library. Pattern numbers can be overwritten or re-used for a different pattern.

ARCFONT *<in_cover> <out_font> <out_pattern>*

FONTARC *<out_font> <pattern_number> <out_cover>*

To display a font library within the *ArcPlot* session, ***FONTDUMP***[2] allows the user to view the font and pattern prior to inserting a new one into the font library. This should prevent overwriting of patterns and decrease confusion as to what pattern numbers are available within that font library.

FONTDUMP *<marker | line | shade> SCREEN*
{start_pattern}{end_pattern}

It is important for the user to keep track of the font and pattern number. This will decrease the likelihood of the user overwriting markers that currently exist. The handling of the fonts and patterns is a user preference and there is no set guideline as to how to manage the font libraries. Often times font libraries are grouped according to usage, for example, all utility markers would exist in one font library.

Arc/Info provides font libraries from 0 to 19, which includes various styles of text, marker patterns, and line styles. Fonts 20 to 25 contain custom patterns which can be altered or manipulated by copying them to a local workspace. This allows for the creation of additional font libraries. Fonts 26 to 40 can be created for further customization of symbology.

Arc/Info searches the local directory first, so any customizing that is done will be used first in place of the fonts in the *IGL* directory, unless otherwise indicated.

The ***FONTCREATE***[2] command at the *ARC* prompt initiates the creation of additional fonts. *C*ustomized fonts should be placed in either the users *local workspace*, the *$ARCHOME/igl63exe/FNT<#>* directory, or *both*.

When *ARC* is executed, the customized fonts will automatically be made available for use within the work session.

FONTCREATE *<font>{xmin}{ymin}{xmax}{ymax}*

Creation of Arc/Info Markersets

Now that the coverage has been placed into a font and assigned a pattern number, creation of a customized markerset begins. Within the *ArcPlot* session the font and the markerset are associated with one another.

Markersets can contain up to 999 symbols. The user needs to keep in mind that the default symbol sets, i.e, Plotter.mrk, Carto.mrk, etc., use marker-symbols 1 to 100. To avoid confusion, it may be easier to start a new markerset beginning with 101. This will eliminate the tendency to overwrite existing markers.

To initiate the creation of a new markerset, the default markerset must be cleared, or it will be read into the selected markerset. In the *ArcPlot* session use the ***MARKERDELETE***[3] command**.** Note: The ***MARKERDELETE*** command can also be used to delete a specific symbol if indicated.

MARKERDELETE *<symbol | all>*

A new markerset can be created by using the ***MARKERSAVE***[3] command. The use of this command will create an empty markerset to hold new markers. Note: The ***MARKERSAVE*** command is also used to save an already existing markerset.

MARKERSAVE *<new_markerset.mrk>*

The new markerset needs to be selected with the ***MARKERSET***[3] command. This command designates the markerset created by using the ***MARKERSAVE*** command above.

MARKERSET *<markerset.mrk>*

A font must be designated with the ***MARKERFONT***[3] command, and the pattern with the ***MARKERPATTERN***[3] command**.**

MARKERFONT *<font_number>*
MARKERPATTERN *<pattern_number>*

Marker color, size, angle, offset, pen, for example, as well as other marker characteristics, can be added to the marker depending on the user's desired output.

As stated previously, text can be added to a symbol using default textsets or custom designed textsets by creating additional layers within the marker. The ***MARKERLAYER***[3] command allows for multiple layers within one marker. ***MARKERLAYER*** specifies which layer in the currently selected marker will be affected by any changes. Placing text from an already existing font library would follow the same procedures mentioned above. The font and pattern must be specified per layer. Multiple layering is NOT limited to text. ***MARKEREDIT***[3] allows interactive modification of markers and markerset files through a series of invoked menus. ***MARKEREDIT*** is very helpful for modifying custom symbols such as editing, saving, deleting, copying and many other drawing requirements. Text can easily be added and manipulated within ***MARKEREDIT***. However, keep in mind that multiple layers are stored in multiple patterns and often times multiple fonts. In the event that these markers need to be translated BACK to *MicroStation* through ***ARCIGDS***[3], the ***FONTARC*** command pulls out ONE pattern at a time. The marker itself is made of multiple layers but is not stored as one element. The output of the marker is based on the referencing of multiple fonts and patterns.

MARKERLAYER <layer>
MARKEREDIT
ARCIGDS <in_cover><out_igds_file><COMPLEX|NOCOMPLEX>
<VANILLA | ACODE | SOURCE> <in_igds_seed_file>
{in_cell_library}{in_property_map}{in_features}{out_elements}

When creating a new marker, it is important to view the marker prior to putting it into the markerset. This is to ensure that the parameters chosen were executed correctly. Use the ***MARKER***[3] command to view the marker. If the marker is satisfactory, it can be placed in the markerset using the ***MARKERPUT***[3] command. The final step in creating a markerset is saving it. This is VERY important - do not exit *ArcPlot* prior to executing the ***MARKERSAVE***[3] command.

MARKER *<* | *xy>*
MARKERPUT *<symbol>*
MARKERSAVE *<markerset.mrk>*

Note: The user can view the newly created markerset using the following command:

SYMBOLDUMP[3] *<line | marker | shade | text> SCREEN*
<start pattern #> <end pattern #>

Once the ***MARKERPUT*** command is executed, the above steps can be repeated for each new symbol. When to save the markerset is at the user's discretion and can be done at any time.

Once the markerset is complete, exit from the *ArcPlot* session and begin with the creation of additional markers as needed.

Conclusion

Here at the Tri-Service CADD/GIS Technology Center, many individuals provide support for the development of the Tri-Service Spatial Data Standards (TSSDS). One of our primary functions is to support the manipulation, modification and creation of symbology in *Arc/Info, MicroStation* and *AutoCAD*. Symbology is gathered from a variety of existing symbols used by many installations and agencies. We also custom design symbols as the need arises. While researching the integration of symbology from one CADD/GIS platform to another, we found that the documentation for creation of symbols, provided by the Environmental Systems Research Institute (ESRI), was very good. However, we needed to integrate the use of other data formats into these methods. Granted, there exists supporting documentation for those methods as well. But our goal was to bring these methods together in an attempt to simplify these concepts. Given the amount of data sharing across various CADD and GIS platforms, we wanted to create a simplified methodology for symbol translation here within The Center. We wanted also to share our methods with the rest of the CADD/GIS community that may be creating symbols in this manner.

References Cited

1. Tri-Service CADD/GIS Technology Center; "The Tri-Service Spatial Data Standards, Release 1.4", 1995. The product was a software application developed to store, manage, and output geospatial data standards. The application was released on CD-ROM, and is available over the Internet from: "http://mr2.wes.army.mil".

2. Bentley Systems, Inc., and Intergraph Corporation, "MicroStation 2D Graphics Level 1, Course Guide", 1994. *MicroStation* is a registered trademark of Bentley Systems, Inc. IGDS is a trademark of Intergraph Corporation.

3. Environmental Systems Research Institute, Inc., "On-Line Documentation: *Arc; ArcPlot; ArcEdit; Map Display, Arc/Info Query and Output*" *Arc/Info* release 7.0.3", 1994. *Arc/Info* is a registered trademark of Environmental Systems Research Institute, Inc.

* Special thanks to the Tri-Service CADD/GIS Technology Center in Vicksburg, Mississippi for their support of this paper and presentation.

* Special thanks given to the technical support staff at Environmental Systems Research Institute, Inc., in Redlands, California, Boston, Massachusetts., and San Antonio, Texas.

APPENDIX

Sample AML's used in the creation

of the Arc/Info Symbology

Note: These AML's have been modified to reflect general path names.

/* ***EDIT.AML*** : THIS WAS USED AFTER EXTRACTION OF MICROSTATION CELLS AND /*THE CREATION OF A DGN. WE ARE UPDATING MARKERS FOR ENTRY INTO A FONT /*LIBRARY.

/* Jim Hoff and Kathy Riegelmann 10/6/95

```
&ECHO &ON

ae
display 9999
ec [getcover * -all]
de arc tic id
draw

/* This calculates the TIC ID's into their proper order. The TIC's are
/* improperly translated from Microstation into Arc/Info. A value of 10
/* is added to each tic. Arc/Info will not allow duplicate tic ids.

sel all
cal $id = $id + 10

sel $id = 11
cal $id = 3

sel $id = 12
cal $id = 1

sel $id = 13
cal $id = 2

sel $id = 10
cal $id = 4

de TIC ID
```

/* ***EDIT.AML***

/* In order to place a coverage into a font, the arcs must be calculated
/* to either 2 (for single lines), 6 (fat lines) and 7 (solid fill).
/* Change the $id to reflect your pattern of choice.

```
ef arc
sel all
cal $ID = 2, 6 or 7

draw

/*  Labels are not necessary for processing of the pattern.

ef label
sel all
delete

/*  This will eliminate the 'block' that was created and translated during
/*  the IGDSARC command. This arc is not needed.

ef arc
sel one
delete

save
q

&ECHO &OFF
```

```
/* MARKER_DF.aml - PUTS SYMBOLS BY FONT AND PATTERN
/*                 INTO A MARKER SET

/*  Jim Hoff and Kathy Riegelmann 10/6/95

&ECHO &ON

ap
disp 9999 3

markerdelete all
markerset [response 'enter marker set'].mrk
markerfont [response 'enter font']
markerpattern [response 'enter pattern']
markercolor [response 'enter color']

/* to view symbol at a larger size

markersize [response 'enter symbol size (for viewing only)']
marker *
markersize [response 'enter size of marker']
markerput [response 'enter markersymbol']
symboldump marker screen {start symbol} {end symbol}
markersave [response 'enter markerset to save'].mrk

yes

quit

&ECHO &OFF
```

```
/*  MARKERINFO.aml:  USED TO DISPLAY MARKER INFO AND CREATE A
/*                   WATCH FILE FOR DOCUMENTATION

/*                    Kathy Riegelmann and Jim Hoff   10/95

ap

&echo &on

markerdelete all

/*  setting a variable for the selected symbolset

        &setvar LIBRARY [response 'Markerset Name']

/*  creating a watch file for storage of markerinfo reference

        &watch %LIBRARY%.wat

markerset %LIBRARY%

&do i := 101 &to 500

MARKERINFO %i%

&end
&return
q

&watch &off

&echo &off
```